Project 3: Supervised Learning

Iterative Dichotomiser 3 (ID3) is an algorithm to construct a decision tree. Decision trees can predict the classification of an object given a set of attributes belonging to that object. Training data is needing to initially form the decision tree. ID3 creates a decision tree by recursively testing each attribute to determine the best split for classification. This is done by calculating the entropy (uncertainty) of the classification for each possible split point for all attributes. Then the information gain is calculated by subtracting the remaining entropy from the initial entropy. The split point with the highest information gain (which will also have the smallest entropy) is chosen. This process is repeated on subsets of the original data until (1) the subset only contains objects of one classification (entropy of zero), (2) there are no more possible split points, or (3) splitting the data does not decrease the entropy. When the data can no longer be split to decrease uncertainty, a leaf node is formed. The leaf node contains the classification of the object with the maximum number of occurrences within its current subset. From here, the decision tree can be used to guess the classifications of other objects. This is done by descending through the tree according to the object's attributes and choosing the classification of the resultant leaf node.

Figure 1 illustrates the ID3 method using a simple dataset with three attributes and two classification options. The figure shows the decision tree that would result from using the ID3 algorithm on the given data. The first split occurs along the color attribute. This is because it was determined that splitting along that attribute decreased the uncertainty of classification the most (the equation for calculating the remaining uncertainty will be discussed later). In facts, all green objects had the classification '-' and all blue objects had a classification of '+'. This allowed for the creation of two leaf nodes because objects that have these attribute values have no uncertainty in classification. However, red objects still have uncertainty with regard to classification. The best split point for the subset of objects with a color attribute of red must be found next. The best split for the subset is a split along the size attribute. This split results in no uncertainty as all big objects in the subset are classified as '+' and all small objects are classified as '-', so two more leaf nodes are created. At this point, the tree is complete, and the algorithm terminates.

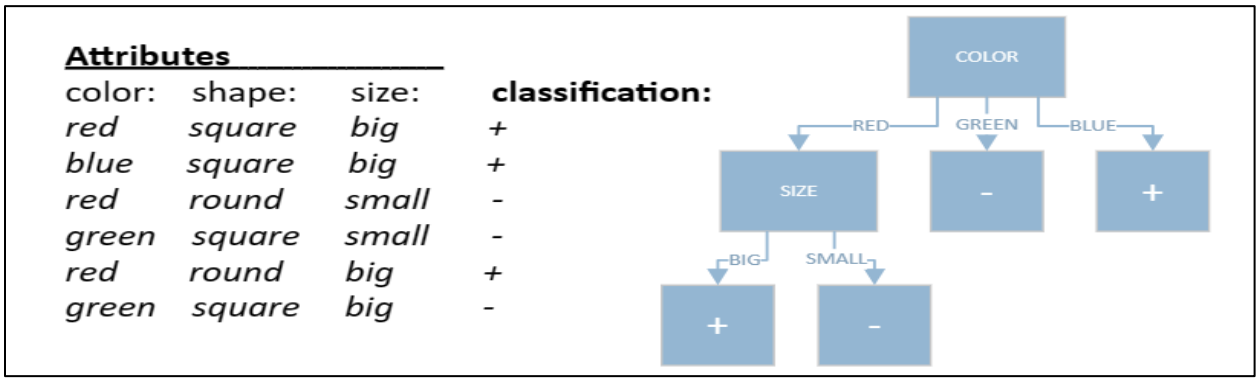


**Figure 1:** The ID3 method.

The code I developed implements the ID3 algorithm to classify objects with continuous attribute values. To begin, the initial entropy is calculated by finding the number of unique objects in the data, the number of occurrences of each unique object, and from there

calculating uncertainty based on the equation Entropy = $\Sigma_i P(C_i)*\log_2(P(C_i))$, where $P(C_i)$ is the probability of selecting the object $C_i$ from the data. After the initial entropy is calculated, the program iterates through the sorted data and splits the data along each possible split point within the attributes (the data will be sorted along each attribute in turn). For example, for each attribute, any time the value of the attribute increases (e.g., from 4.3 to 4.4), the data is split, and the resultant entropy is calculated. This is done by summing the entropy in each subset. To find the entropy in each subset, first find the probability of selecting the current subset from the data as a whole (the two current subsets summed) (call this P(A)). Next, find the number of unique objects within the subset and the number of occurrences of each unique object. Finally, calculate uncertainty based on the equation Entropy = P(A) * ($\Sigma_i P(C_i)*\log_2(P(C_i))$), where $P(C_i)$ is the probability of selecting the object $C_i$ from the subset and P(A) is the probability of selecting the current subset from the data. The information gain is calculated by subtracting the subsets' entropy from the original entropy. As the program iterates through each attribute, the split point with the highest information gain is saved and chosen as the final splitting point of the data. The function performing the ID3 algorithm recursively calls itself with subsets of the data to calculate all the children nodes by finding all the best split points. As mentioned before, when the data can no longer be split to decrease uncertainty, a leaf node is formed based on the classification of the object with the maximum number of occurrences within the current subset.

After the decision tree has been trained on the training data, it is tested on the validation data. To classify each object in the validation data, the program descends through the decision tree, picking children according to the object's attributes. When a leaf node is reached, the program classifies the object according to the leaf node's classification. If the classification was correct, the program adds one to the final number of correctly classified objects. This number is displayed after the program has finished classifying all the objects in the validation data.

After completing the program, I ran tests to determine the accuracy of the classification under a variety of circumstances. I tested the program on two separate datasets, one on iris classification (iris-data.txt) and one on cancer classification (cancer-data.txt). The iris data includes four attributes (sepal length, sepal width, petal length, and petal width) and the actual classification of the iris (Iris Setosa, Iris Versicolour, or Iris Virginica) which are labeled zero through two. The cancer dataset is focused on breast cancer. It includes nine attributes and a final breast cancer classification (this can be from zero to five). The iris data file has one hundred and fifty rows (150 objects). To test the accuracy of the program under different circumstances, I separated random objects to be used for training and validation and ran the program one hundred times for randomly selected training and validation sets of the same size. For the iris data, I performed this process with validation set of sizes 1, 5, 10, 25, 50, 75, 100, 125, 140, 145, and 149. The cancer dataset has one hundred and five rows (105 objects). I repeated the same process with the cancer data but with validation sets of sizes 1, 5, 10, 25, 50, 75, 90, 100, 104. I then calculated the mean and standard error of the mean of the percentage of testing examples correctly classified by my decision trees for each data set. The results for the iris dataset are in Table 1 and the results for the cancer dataset are in Table 2. Additionally, I plotted the Mean ± (1.96 * (Standard Error of the Mean)) across all validation set sizes. The results for the iris dataset are in Figure 2 and the results for the cancer dataset are in Figure 3.

| V | 1 | 5 | 10 | 25 | 50 | 75 | 100 | 125 | 140 | 145 | 149 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **Mean** | 94.00 | 93.00 | 93.60 | 93.52 | 94.06 | 94.37 | 94.06 | 90.98 | 76.21 | 62.01 | 32.89 |
| **σ** | 2.375 | 1.179 | 0.831 | 0.411 | 0.260 | 0.217 | 0.241 | 0.494 | 1.175 | 1.132 | 3.6E-15 |

**Table 1:** Results of the iris dataset, where σ is the standard error of the mean and V is the number of validation examples.

| V | 1 | 5 | 10 | 25 | 50 | 75 | 90 | 100 | 104 |
|---|---|---|---|---|---|---|---|---|---|
| **Mean** | 75.00 | 67.40 | 68.90 | 67.04 | 61.76 | 59.53 | 53.38 | 38.89 | 16.74 |
| **σ** | 4.330 | 1.911 | 1.378 | 0.878 | 0.665 | 0.566 | 0.701 | 0.844 | 0.310 |

**Table 2:** Results of the cancer dataset, where σ is the standard error of the mean and V is the number of validation examples.

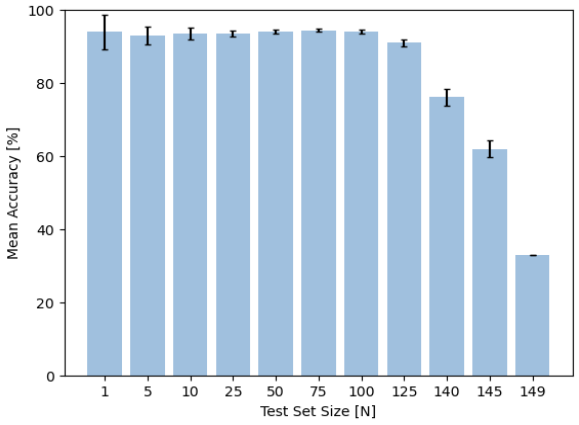


**Figure 2:** Plot for the iris dataset

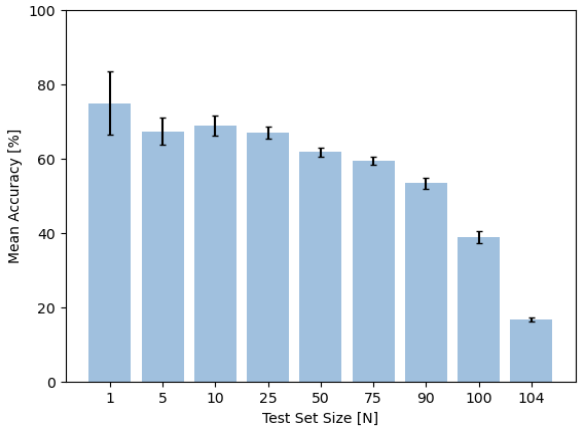


**Figure 3:** Plot for the cancer dataset

For the iris dataset, validation set sizes from one to one hundred and twenty-five perform fairly similarly (and better than the validation sizes from 140 to 149). The smaller the size of the validation set, the larger the training dataset. This indicates that it is important for the decision tree to be trained on a significant portion of the data in order to have a high accuracy count. The cancer dataset follows a similar trend. Validation set sizes from one to twenty-five perform fairly similarly (and better than the validation sizes from 50 to 104). One thing that is not well illustrated here is the importance of not overtraining the data. Sometimes, the decision tree can be overly meticulous in dividing the data according to the training data and thus creates divisions that are not widely beneficial, only applying to the training data specifically. However, it is interesting to note that overtraining did not seem to be a problem with these examples, as the decision trees for both datasets that were trained on almost all of the data did well. There were also differences in how well the decision trees classified the data in the two datasets. The generated decision trees consistently did better at classifying the iris data. Two differences in the datasets that may be the cause of this disparity are the number of attributes and the total number of classifications. The iris dataset only had four attributes and a total of three classifications, whereas the cancer dataset had nine attributes and six total classifications. The results of both datasets indicate the importance of carefully choosing the sizes of the training and validation sets to maximize the overall performance of the decision trees.